

SN3262 – Network Administration, Management & Security

Instances of MIB Objects, Lexicographical Ordering & Systems, Interface, and IP MIB objects for Network Management Reading:

rfc 1155, 1157, rfc1213, rfc2011 and rfc2863

Stallings, W (1999) SNMP, SNMPv2, SNMPv3, and RMON1 and 2. 3rd ed. Reading Massachusetts, Addison-Wesley.

Leinwand, A. & Conroy, K. F. (1996) Network Management: A Practical Perspective 2nd ed. Addison-Wesley. Chapter 10.

http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm

<http://www.oid-info.com/index.htm>

<http://www.et.put.poznan.pl/snmp/main/mainmenu.html>

All the above accessed 30/10/08.

Object Types and Instances

An object type is a definition of a kind of managed object.

An object instance is an instantiation of an object type which has been bound to a value.

E.g. the notion of an entry in a routing table might be defined in the MIB. Such a notion corresponds to an object type; individual entries in a particular routing table which exist at some time are object instances of that object type.

Instance Identification

Every object in the MIB has a unique object identifier, which is defined by the position of the object in the tree-structured MIB.

When an access is made to a MIB, via SNMP or some other means, it is a specific instance of an object that is wanted, not an object type. This distinction is essential for objects that appear in tables (called columnar objects). For such objects the object identifier alone is not enough to identify the instance.

There is one instance of each object for every row in the table. Therefore we need some convention by which a specific instance of an object within a table may be identified. Reference to object instances is protocol-specific. It is not defined in the MIB.

For scalar objects, there is no ambiguity between the object type and an instance of that object (one-to-one relationship). For consistency with tabular objects, and to distinguish between an object type and an object instance, SNMP dictates that the instance identifier of a scalar object consists of its object identifier concatenated with 0.

For example, suppose one wanted to identify an instance of the variable sysDescr. The object class for sysDescr is:

iso	org	dod	internet	mgmt	mib	system	sysDescr
1	3	6	1	2	1	1	1

Hence, the object type, x, would be 1.3.6.1.2.1.1.1 to which is appended an instance sub-identifier of 0. That is, 1.3.6.1.2.1.1.1.0 identifies the one and only instance of sysDescr.

From rfc 1157

Instance Identification in SNMP

Serial-access technique

This is based on a lexicographic ordering of objects (see later).

Random-access technique

An instance of a scalar object of a particular row of a table is the concatenation of:

- the object type identifier of the table object;
- the suffix that identifies a row object;
- the suffix that identifies the scalar element in that row and
- one set of values of the INDEX objects.

Example

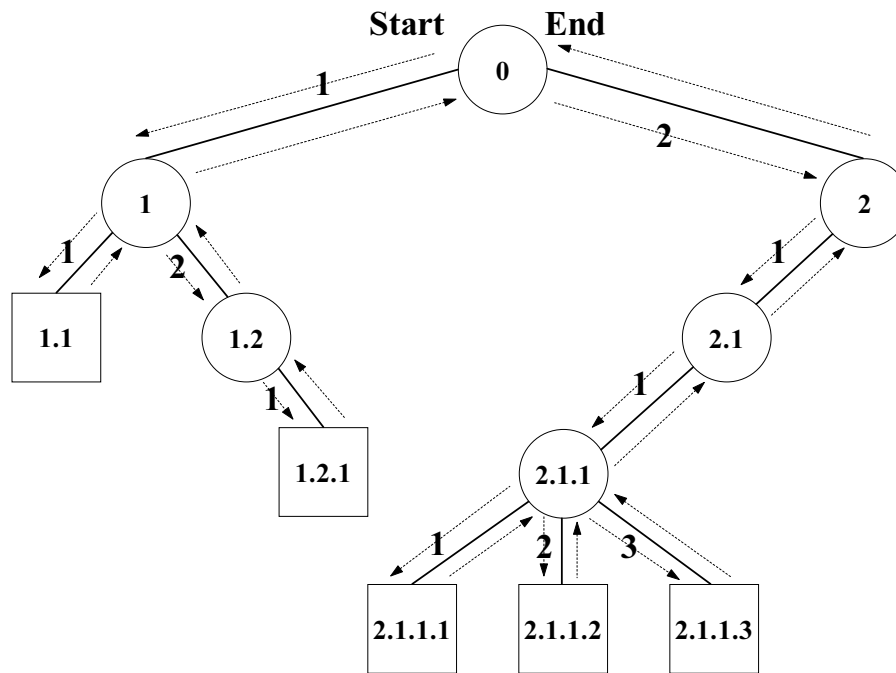
```
tcpConnEntry OBJECT-TYPE
    SYNTAX  TcpConnEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "Information about a particular current TCP
        connection.  An object of this type is transient,
        in that it ceases to exist when (or soon after)
        the connection makes the transition to the CLOSED
        state."
    INDEX   { tcpConnLocalAddress, tcpConnLocalPort,
              tcpConnRemAddress, tcpConnRemPort }
    ::= { tcpConnTable 1 } ← 1.3.6.1.2.1.6.13.1

TcpConnEntry ::= SEQUENCE { tcpConnState INTEGER,
                             tcpConnLocalAddress IpAddress,
                             tcpConnLocalPort INTEGER (0..65535),
                             tcpConnRemAddress IpAddress,
                             tcpConnRemPort INTEGER (0..65535) }
← 1.3.6.1.2.1.6.13.1.1
```

The connection state of the connection indexed by (149.170.13.8, 45010, 149.170.13.7, 22) will be identified by 1.3.6.1.2.1.6.13.1.1.149.170.13.8.45010.149.170.13.7.22

Lexicographical Ordering

An object identifier is a sequence of integers that defines the location of the object in the MIB tree structure. The sequence of integers exhibit a lexicographical ordering. That ordering corresponds to traversing the tree of objects identifiers in depth-first mode with child nodes of a common parent depicted in ascending numerical order. This ordering extends to object instance identifiers. An ordering is important when the manager does not know the exact makeup of the MIB view that an agent presents to it. By using the get-next operation, the SNMP management station can ask the next object in that ordering. It works even if the supplied identifier is not valid, i.e. does not exist in the MIB. In that case, this is the next valid identifier that is returned. It is also useful to access tables row by row.



Lexicographical ordering in an object identifier tree can be generated as follows:

The tree must be drawn so that the branches are arranged in increasing order left to right.

The lexicographical order is generated by traversing the tree in what is known as *pre-order traversal* or *depth-first search*, which is defined as follows:

- Visit the root
- Traverse the subtrees from left to right

What is the lexicographical ordering of all of the nodes in the above tree?

What is the lexicographical ordering of all of the leaf nodes in the above tree?

Management Areas for System Group Objects

Object	Management Area	Object	Management Area
sysObjectID	Fault	sysDescr	Configuration
sysServices	Fault	sysLocation	Configuration
sysUptime	Fault	sysContact	Configuration
		sysName	Configuration

Management Areas for Interfaces Group Objects

Object	Management Area	Object	Management Area
ifAdminStatus	Fault	ifInOctets	Performance
ifOperStatus	Fault	ifInUcastPkts	Performance
ifLastChange	Fault	ifInDiscards	Performance
ifTestTable*	Fault	ifInErrors	Performance
ifDescr	Configuration	ifInUnknownProtos	Performance
ifName	Configuration	ifOutOctets	Performance
ifType	Configuration	ifOutUcastPkts	Performance
ifMTU	Configuration	ifOutDiscards	Performance
ifSpeed	Configuration	ifOutErrors	Performance
ifAdminStatus	Configuration	ifInMulticastPkts*	Performance
ifHighSpeed*	Configuration	ifInBroadcastPkts*	Performance
ifPromiscuousMode*	Configuration	ifOutMulticastPkts*	Performance
ifLinkUpDownTrapEnable*	Configuration	ifOutBroadcastPkts*	Performance
ifRcvAddressTable*	Configuration		

* not defined in rfc1213, defined in later rfcs including rfc2863

Management Areas for IP Group Objects

Object	Management Area	Object	Management Area
ipForwarding	Configuration	ipInUnknownProtos	Performance
ipAddrTable	Configuration	ipInDiscards	Performance
ipRouteTable	Configuration	ipInDelivers	Performance
ipInReceives	Performance	ipOutRequests	Performance
ipInHdrErrors	Performance	ipOutDiscards	Performance
ipInAddrErrors	Performance	ipOutNoRoutes	Performance
ipForwDatagrams	Performance		

Examples of use of MIB Objects

Example 1 - Percentage of input and output errors on an interface

$$\begin{aligned}\text{total packets received} &= \text{ifInUcastPkts} + \text{ifInBroadcastPkts} + \text{ifInMulticastPkts} \\ \text{total packets sent} &= \text{ifOutUcastPkts} + \text{ifOutBroadcastPkts} + \text{ifOutMulticastPkts} \\ \text{Percentage input errors} &= \frac{\text{ifInErrors}}{\text{total packets received}} \times 100 \\ \text{Percentage output errors} &= \frac{\text{ifOutErrors}}{\text{total packets sent}} \times 100\end{aligned}$$

Example 2 - Percentage utilisation of an interface

This requires two polls, one at time x and one at time y

$$\begin{aligned}\text{total bytes in interval } (y - x) &= (\text{ifInOctets}_y - \text{ifInOctets}_x) \\ &\quad + (\text{ifOutOctets}_y - \text{ifOutOctets}_x) \\ \text{total bytes per second} &= \frac{\text{total bytes}}{y - x} \\ \text{Percentage line utilisation} &= \frac{\text{total bytes per second} \times 8}{\text{ifSpeed}} \times 100\end{aligned}$$

Example 3 - Percentage IP input and output errors

$$\begin{aligned}\text{Percentage IP input errors} &= \frac{\text{ipInDiscards} + \text{ipInHdrErrors} + \text{ipInAddrErrors}}{\text{ipInReceives}} \times 100 \\ \text{Percentage IP output errors} &= \frac{\text{ipOutDiscards} + \text{ipOutNoRoutes}}{\text{ipOutRequests}} \times 100\end{aligned}$$

Example 4 - IP forwarding and input rates

$$\begin{aligned}\text{IP forwarding rate} &= \frac{\text{ipForwDatagrams}_y - \text{ipForwDatagrams}_x}{y - x} \\ \text{IP input rate} &= \frac{\text{ipInReceives}_y - \text{ipInReceives}_x}{y - x}\end{aligned}$$